Trees - 6

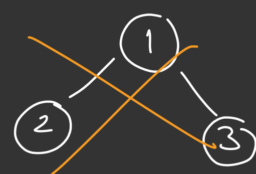
Tree + DP (Popular Interview Problem)

Q) Given N Nodes numbered from 1 to N How many unique BST can be formed using these N Nodes

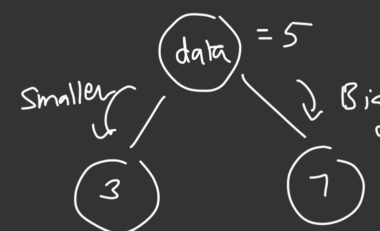
N = 3

output = 5

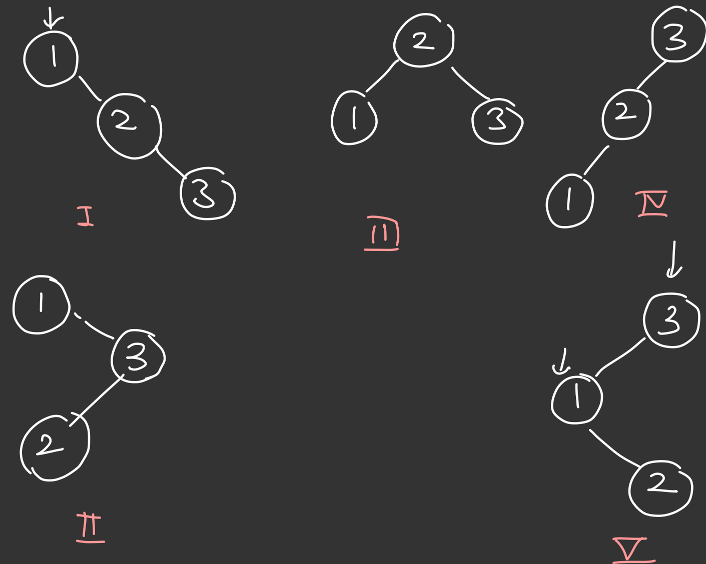
1 2 3



→ BST?



$\boxed{\text{5 BST's}}$

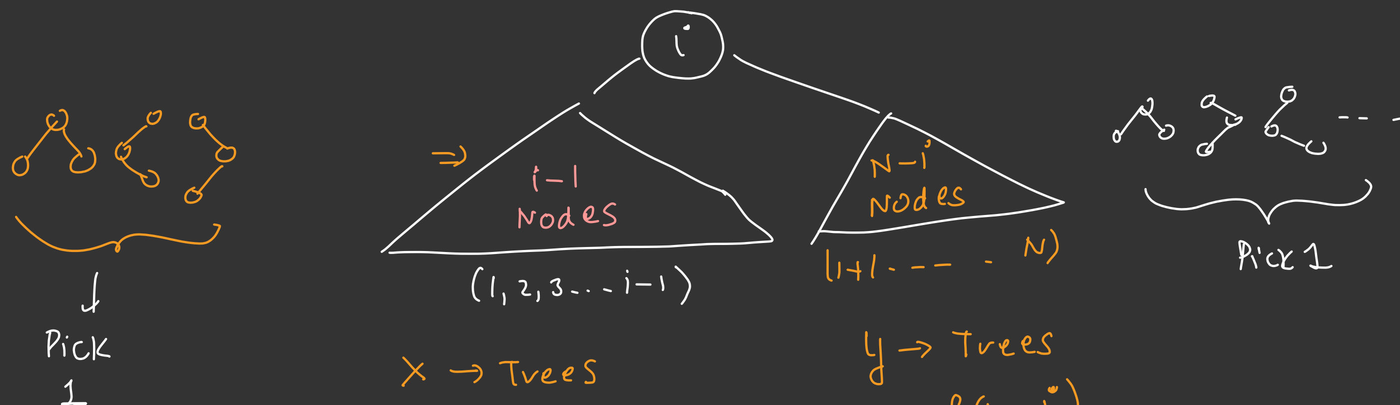


$f(N)$ = how many unique BST's can be formed using all N Nodes.

$$\underbrace{1,\ 2,\ 3,\ 4,\ 5,\ 6\ \ldots\ i-1}_{LS}\ \ i\ \ \underbrace{\ldots\ldots\ N}_{N-i\ \text{Nodes}}$$

Any node can become the root

Let us assume i^{th} Node to be the root



$x \rightarrow$ Trees $= f(i-1)$

$y \rightarrow$ Trees $= f(N-i)$

Left Tree && $\underline{i^{th}\ \text{Node}}$ && Right Tree

$${}^{x}C_1 \quad \cdot \quad 1 \quad \cdot \quad {}^{y}C_1$$

$= x\,y$ ways to build tree with i^{th} Node

$= f(i-1) * f(N-i)$

Total ways =

ans = 0

```
for (i=1, i<=N; i++){

    ans = ans + f(i-1) * f(N-i)

}
```

```
if (n==1)
    return 1
```

$N = 0$ 1   Null Tree

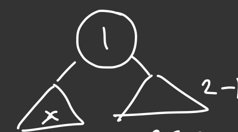
$N = 1$ 1

$N = 2$

1   2

$i=1$ $i=2$

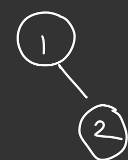
$1 + 1 = 2$

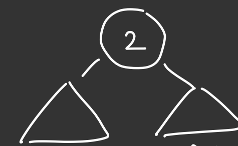


$f(0)$ $f(1)$

$1 * 1$

$= 1$

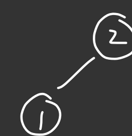




$f(1)$ $f(0)$

$1 * 1$

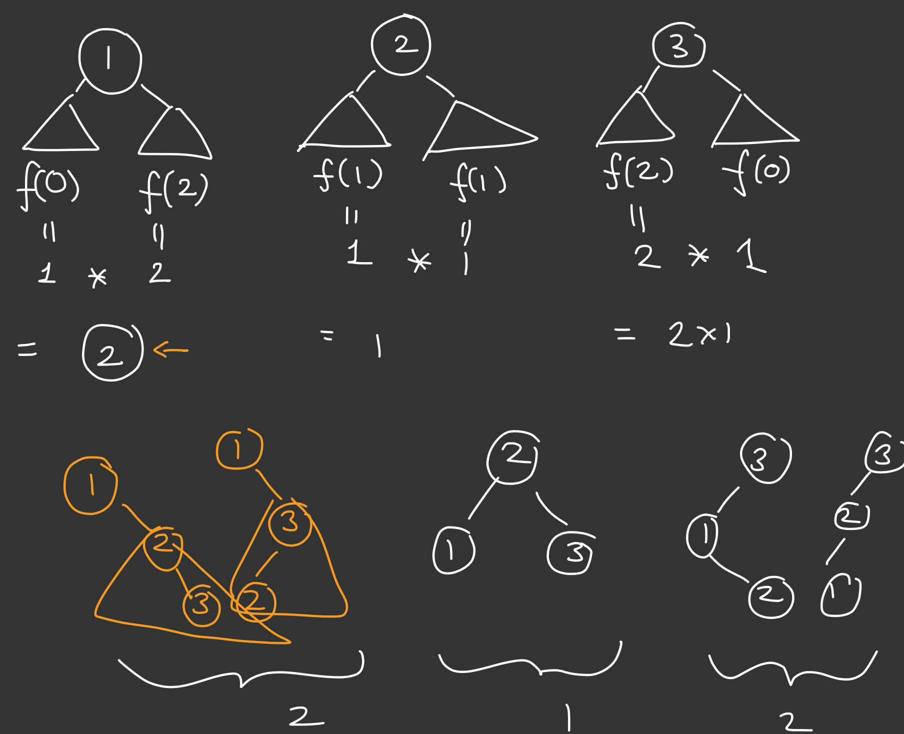
$= 1$



$N=3$ 1 2 3

$2 + 1 + 2$

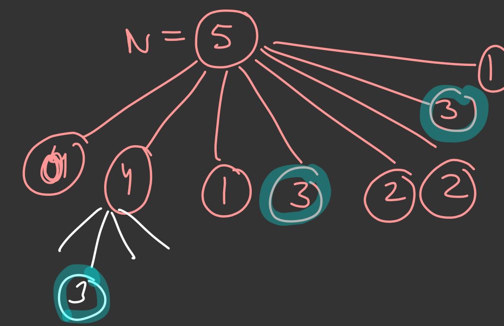
$= \boxed{5}$



```
int f ( int N){

    if (N==0 || N==1)
            return 1
    if(dp(N) is computed) { return dp[N] }
    ans = 0
    for (i=1 to N){

        ans = ans + f(i-1) * f(N-i)

    }
    dp[N]=ans
    return ans;
}
```

smaller states.

i=1

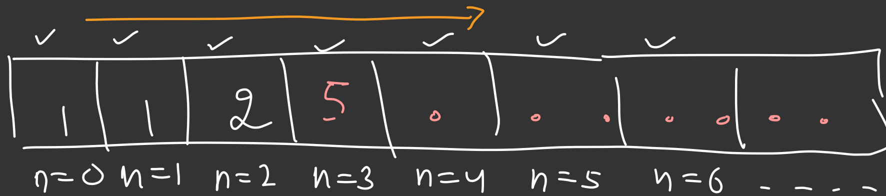


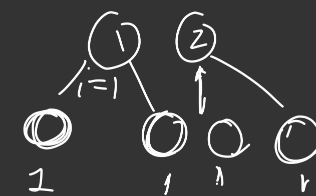
N

1

$O(N!)$

dp[n] = no of BST's possible with N Nodes





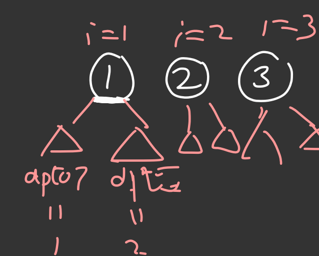
ans = 0 + 1 + 1
= 2

Rec Call

```
for (n=2, n <= N ; n++) {
    ans = 0
    for(i=1 ; i<=n; i++){        // i as root
        ans = ans + dp[i-1] * dp[n-i]
    }
    dp[n] = ans
}
return dp[N]
```

TC :

$O(N^2)$



ans = 2 + 1×1 + 2×1

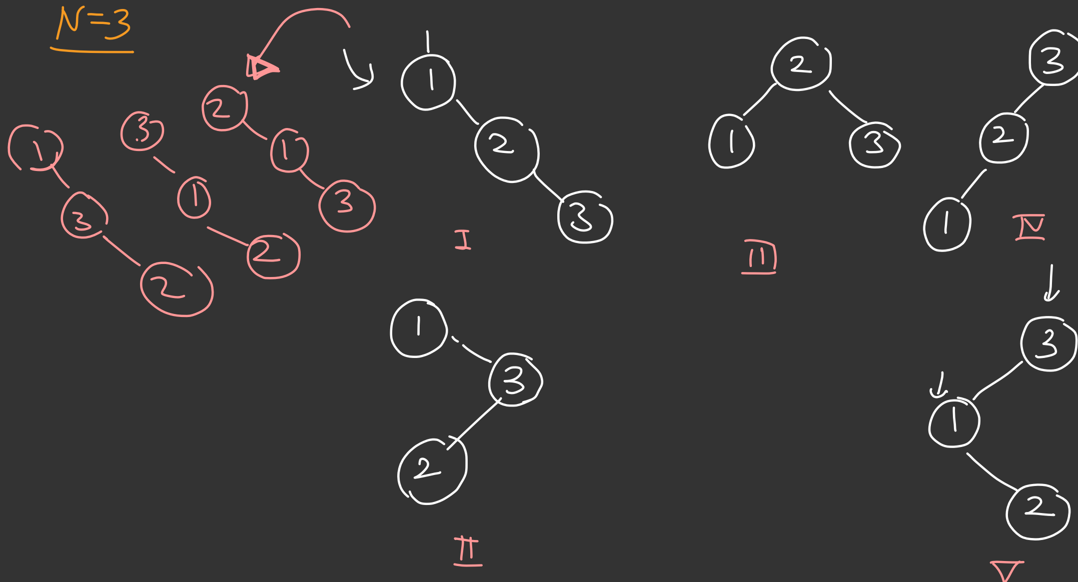
}

= 5

No of Binary Trees with N Nodes.

Hint - elements don't follow any fixed order

N=3



(B1, B2, B3)

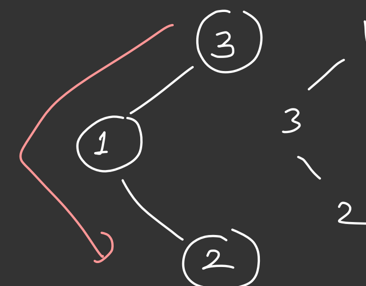
$n! \Rightarrow$ ways to arrange

B1	B3	B2
B1	B2	B3
B2	B3	B1
B2	B1	B3
B3	B1	B2
B3	B2	B1

$3!$

$= 6$

arrangement



$\times\ 3!$ ways.

BST
(1 way)

BT

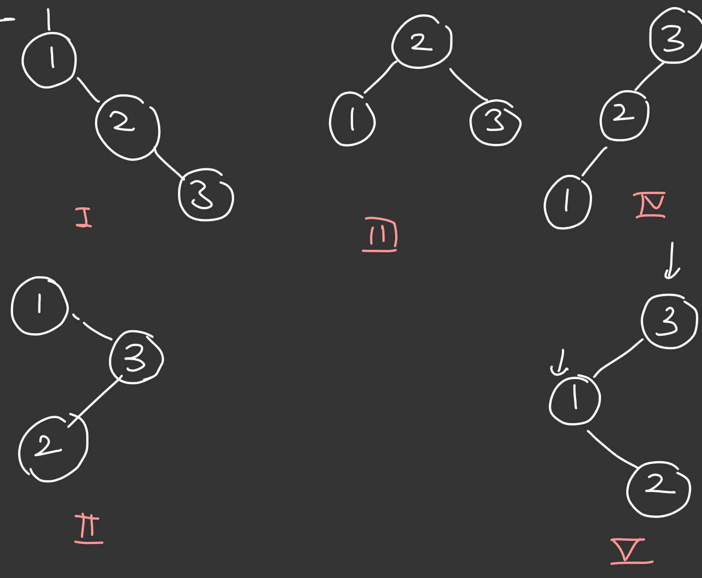
BT ways $=$ $\boxed{(\text{BST ways})}$ $\times\ N!$

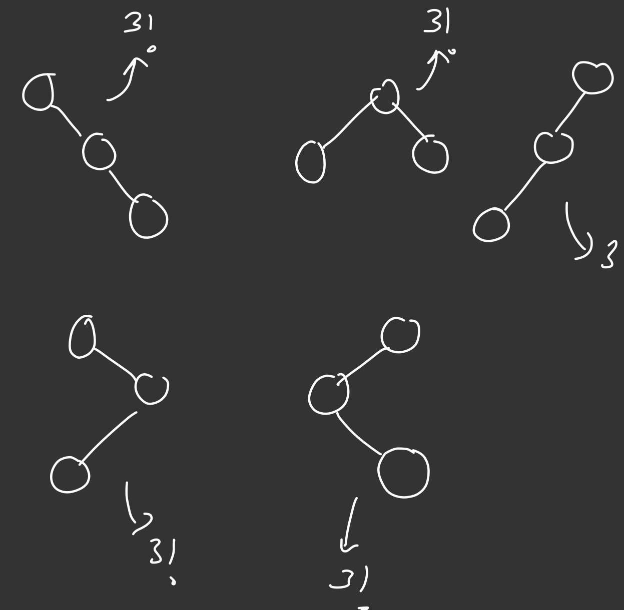
$= (5) \times 3!$

$= 5 \times 6 = $ 30 ways

$N = 3$

N=3





$5 \times 3!$

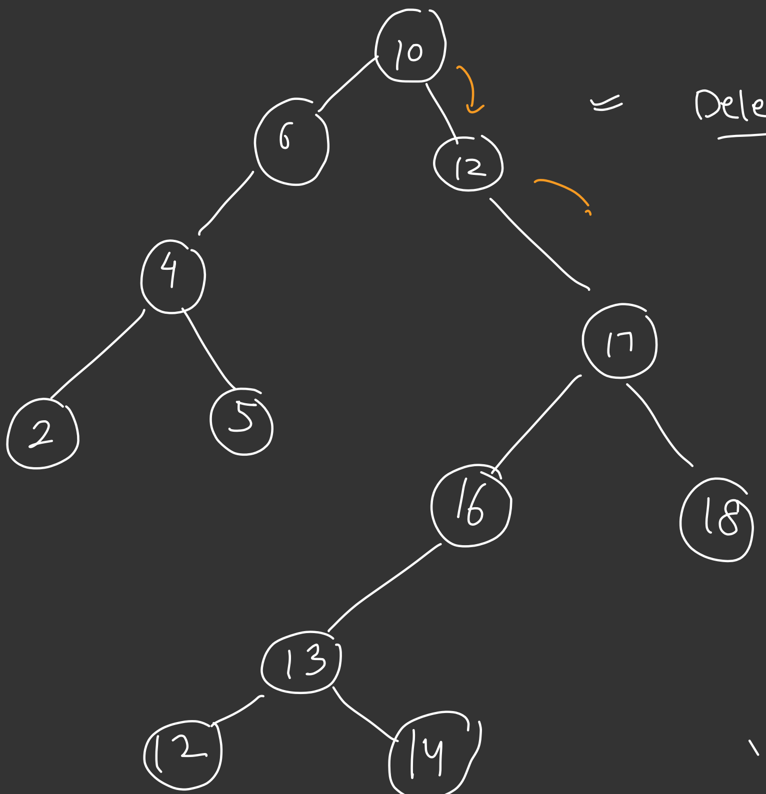
Unique structures = No of BST

Deletion in BST

✓ Insertion $O(\log N) \le O(H) \le O(N)$

✓ Searching

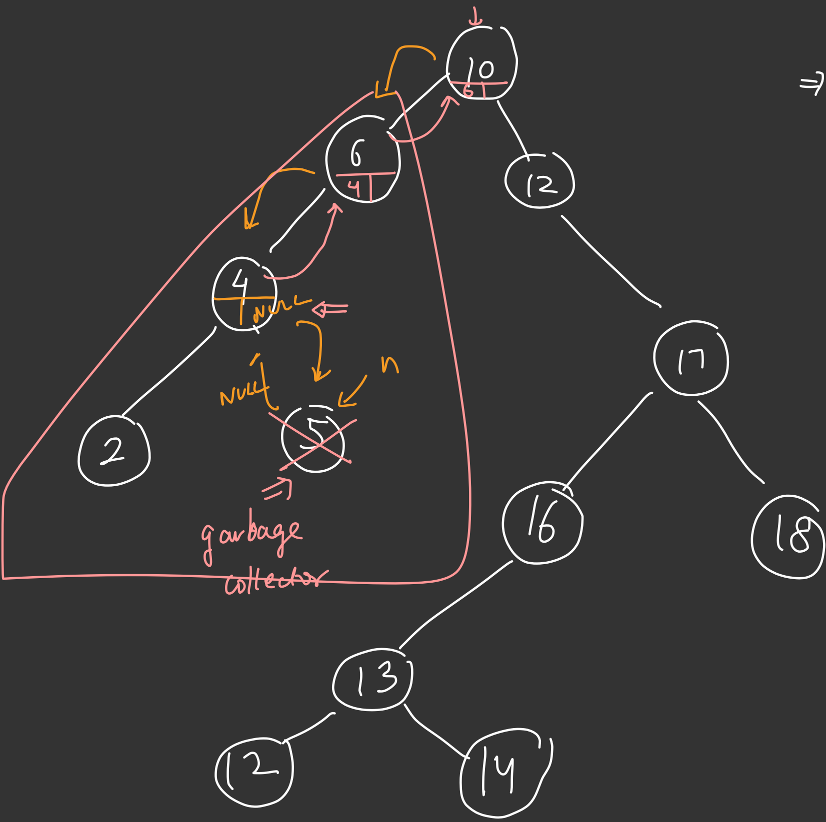
BT	BST
$O(N)$	$O(H)$



✓ Deletion – delete node with data 'x'

Types of Nodes

1. Leaf Nodes / 0 child
2. Single child
3. 2 children



⇒ [No children]

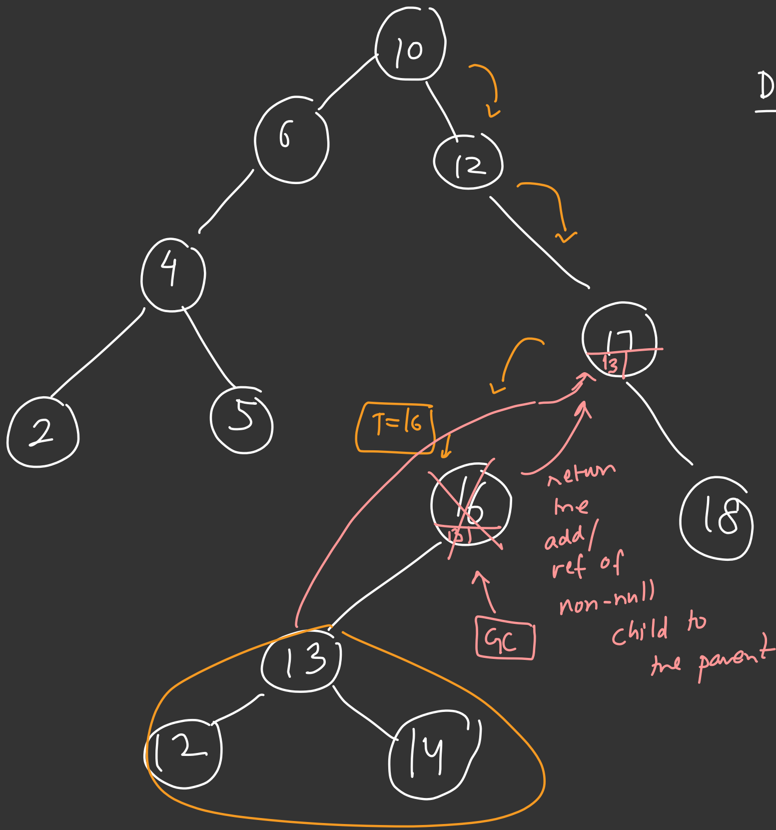
Delete 5

Node delete (root, target)

== target
return NULL

Return root

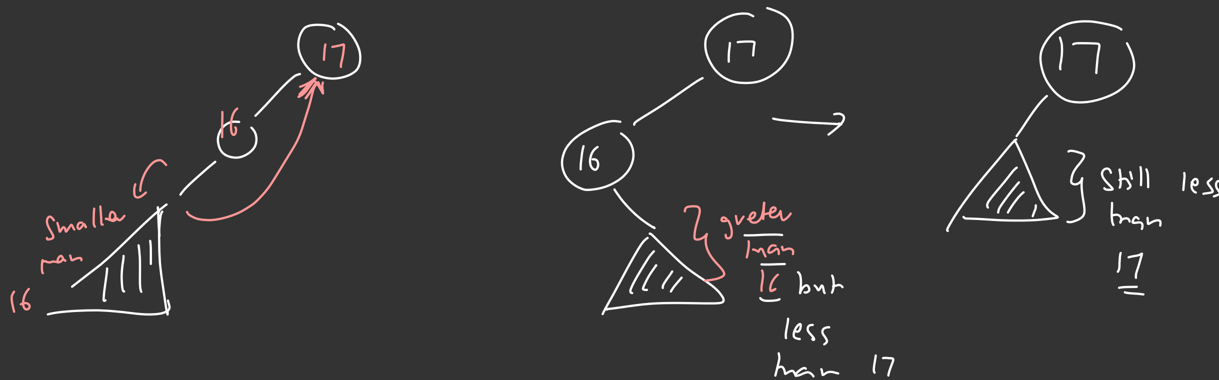
}

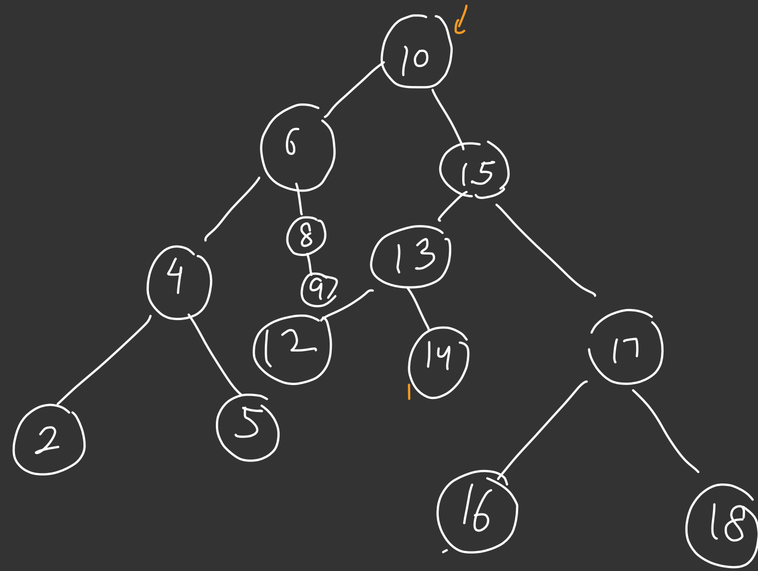


Delete a Node with Single Child

Target = 16

```
if ( root.data == Target ) {

    if (root.left != null ) {
        return root.left
    }
    return root.right

}
```


17
16
smaller than 16
17
16
greater than 16 but less than 17
17
still less than 17



Delete a Node with 2 children

Delete $\boxed{10}$

2, 4, 5, 6, 8, 9 ~~10~~, 12, 13, 14, 15, 16, 17, 18

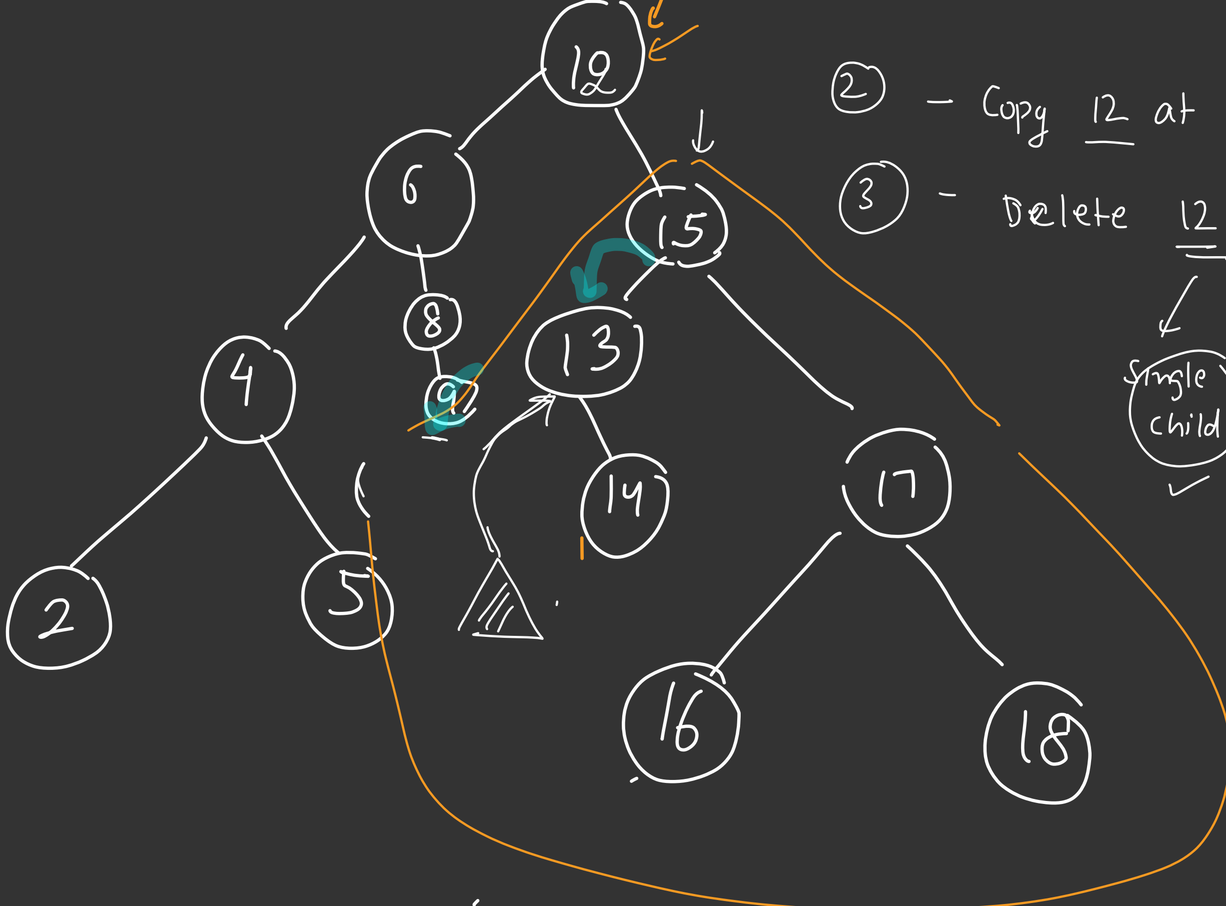
get the next node

largest in LST — 9

or

smallest in RST — 12

1 - find smallest in RST



(2) – Copy 12 at 10's place

(3) – Delete 12 in RST $\boxed{\text{Rec}}$

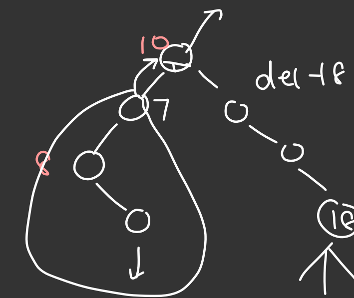
Single child ✓ No child ✓

2, 4, 5, 6, 8, 9, 12, 13, 14, 15, 16, 17, 18

Code

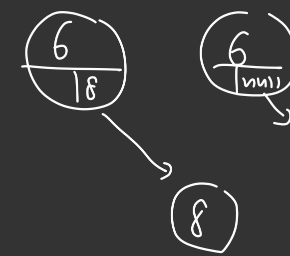
```
Node    deleteNode (Node root, int Key)

  // Search
  if (root == null)
        return null

  // Left          14
  if (Key < root.data) {
      root.left = deleteNode(root.left, Key)
      return root
  }
else if (key > root.data) {
      root.right = delete Node(root.right, Key)
      return root
  }
else {
   // Delete the current 'root' node
```

10:35



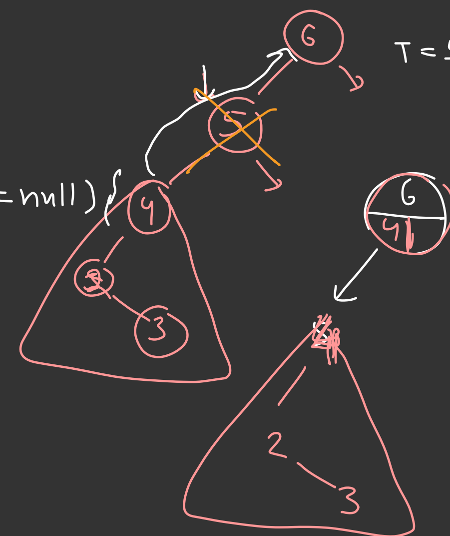
Case I

```
if (root.left == null && root.right == null) {
        return null;
}
```



Case-II

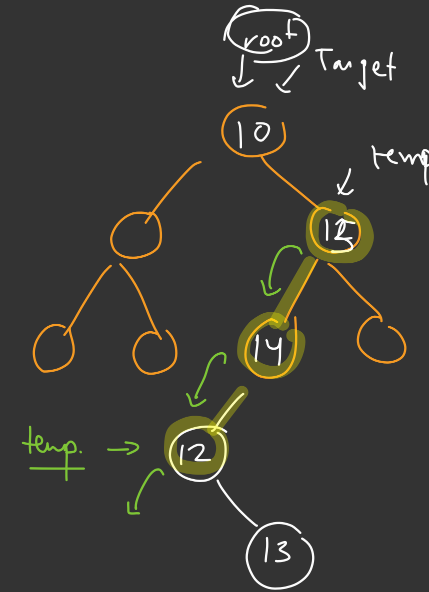
```
if (root.left != null && root.right == null){
        return root.left
}
if (root.left == null && root.right != null){
        return root.right
}
```



Case-III // Two children

Step-1 find smallest node in RST

```
=> temp = root.right
while(temp.left != null){
    temp = temp.left
}
```

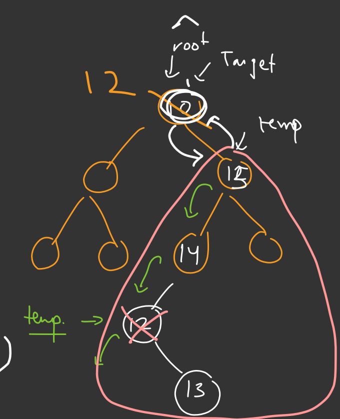


Step-2 Copy Data

```
root.data = temp.data
```

=> Step-3 Delete the redundant in RST

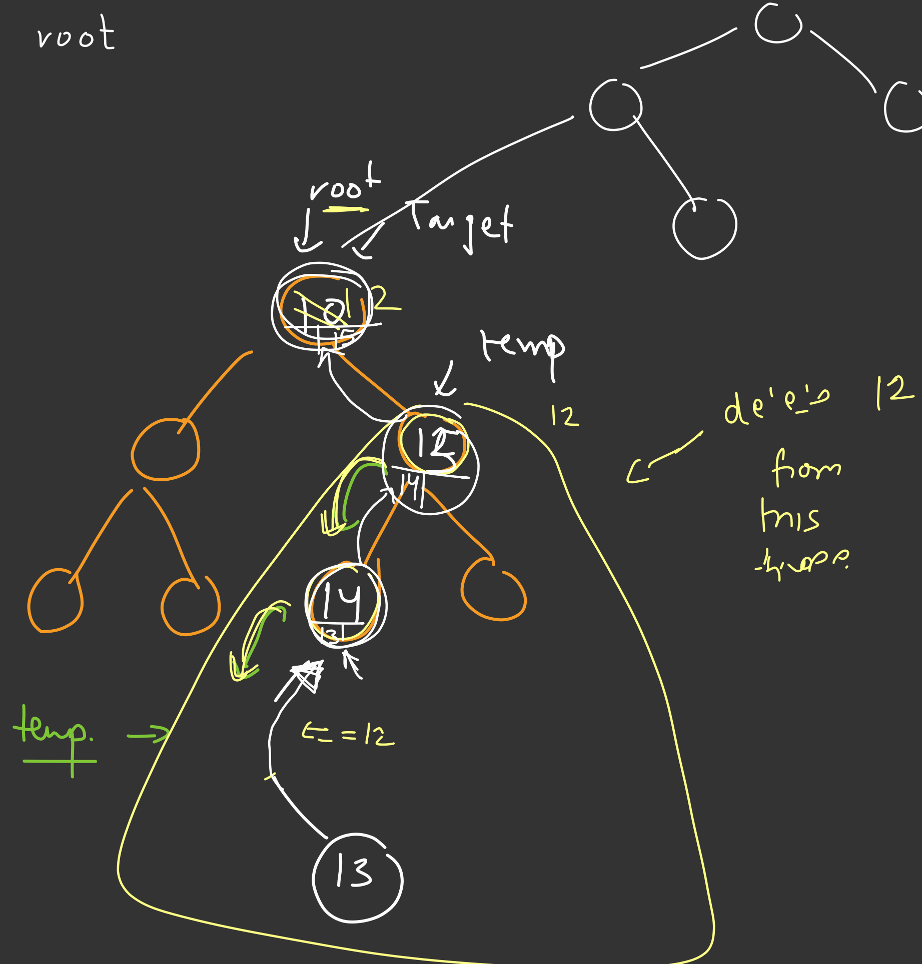
```
root.right = deleteNode(root.right, temp.data)
```



return root

}

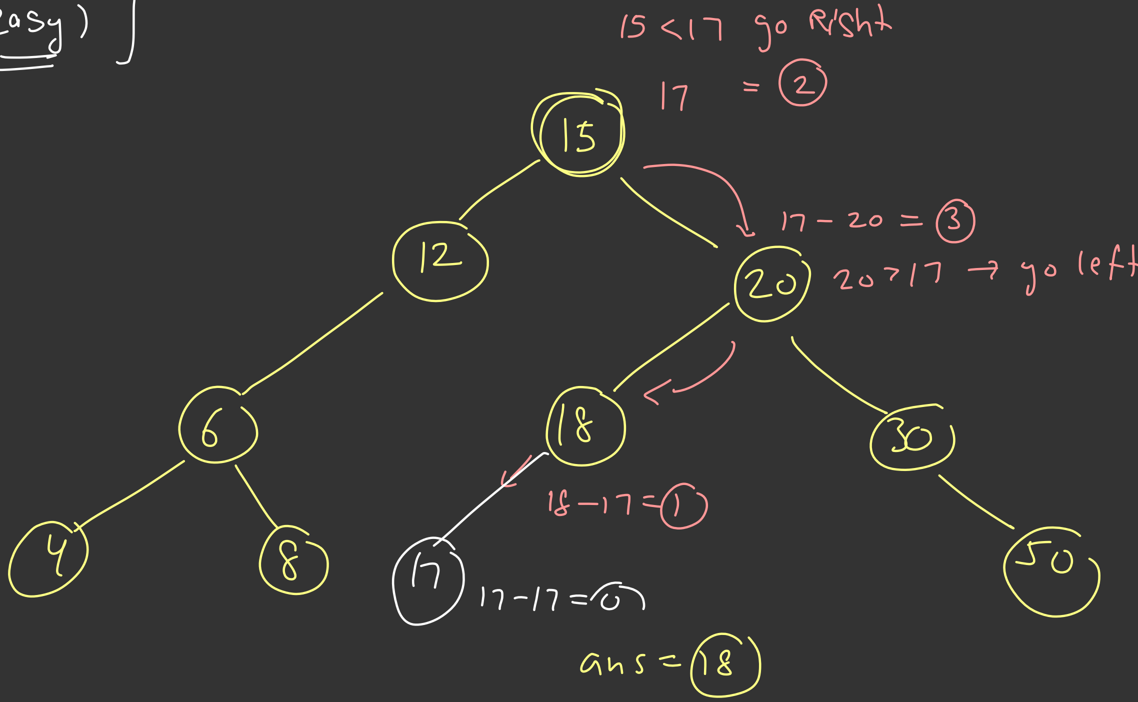
}



Homework

[(easy)]

given a BST, find a node closest to a given Target.



Target = ~~27~~

17

→ go left or Right

→ iteratively

→ track the min diff